Optimización Proyecto:
MICROADAM: Accurate Adaptive Optimization with LowSpace Overhead and Provable Convergence

Aguilera Hernández Edgar Iván
edgar.aguilera@cimat.mx, M. en C.C. CIMAT, Gto

Larralde Ortiz Emmanuel Alejandro
emmanuel.larralde@cimat.mx, M. en C.C. CIMAT, Gto

17 de febrero de 2026

En esta modificación sobre el popular algoritmo de optimización estocástica ADAM, se busca reducir los costos en memoria que suponen las variaciones del método tradicional al momento de estimar el gradiente y sus momentos para cada uno de los parámetros, mientras, tomando como referencia los procesos de compresión y corrección de error descritas en otras publicaciones, en teoría se conserva la tendencia convergente que el resto de método de manejo de memoria eficiente carecen.

1. Marco Teórico

1.1. ADAM tradicional

Como un método de optimización por descenso de gradiente estocástico, ADAM [4] utiliza el gradiente de primer orden así como estimaciones adaptables del primer, y segundo momento (m, v), controlados a través de su respectiva tasa de decremento (β_1, β_2). Este algoritmo combina las ventajas de los métodos *AdaGrad* para lidear con gradientes *sparse*, y RMSProp para trabajar sobre objetivos no estacionarios y en tiempo real (*on-line*), por lo que ADAM apunta a optimizar funciones de altas dimensiones o sobre grandes conjuntos de datos, propios de problemas de aprendizaje máquina.

Estableciendo la diferencia sobre los métodos que utiliza como referencia, aunque similar con respecto al uso de los momentos, ADAM actualiza el valor de sus parámetros utilizando el actual valor del gradiente, en lugar de una versión escalada de este como sucede en RMSProp, adicionalmente,las correcciones que realiza para mitigar el sesgo hacia 0 inducidas por la inicialización de los momentos con vectores de este valor, permiten reducir el riesgo de divergencias que producen los tamaños de paso cada vez mayores si $\beta_2 \to 1$ [4].

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

$$donde:$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

Corrección de sesgo:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

1.2. MicroADAM

Si bien el método tradicional de ADAM estableció como objetivo la optimización de funciones de una gran cantidad de parámetros, los requerimientos de memoria para los modelos actuales con hasta billones de variables (como es el caso de LLM), volvieron a ADAM un algoritmo costoso debido a la estimación de los 2 parámetros adicionales por cada variable en el modelo. Es así que MicroADAM, tomando como referencia los trabajos contemporáneos que buscan conservar las características adaptativas de este optimizador clásico [5][6][7], buscan generar una eficiencia de la memoria requerida (sacrificando ciertas garantías sobre la convergencia para las propuestas desarrolladas hasta ese momento).

Una de las aportaciones de MicroADAM con respecto a estos métodos de reducción de memoria, es que tras la perdida de información sobre el gradiente, como resultado de la proyección en menores dimensiones, el error estimado resultante es utilizado como retroalimentación en las posteriores iteraciones, asegurando una convergencia que no existía en los otros métodos. Aunque este *buffer* de datos perdidos supone un incremento proporcional al tamaño de memoria necesario en el método clásico, MicroADAM también realiza una compresión sobre este error de proyección, consiguiendo así mantener las garantías de convergencia y reducción de memoria necesaria para almacenar los parámetros de estimación.

1.2.1. Compresión de gradiente y error

Para lograr la reducción espacial de estos parámetros, MicroADAM emplea uno de los principales elementos de la optimización distribuida (enfocada al procesamiento descentralizado y en paralelo de grandes conjuntos de datos): los métodos de compresión de gradiente, en el que se intenta mitigar el *trafico* de comunicación que supone compartir esta estimación entre los distintos sub-procesos dedicados a la optimización, dentro de las múltiples técnicas, para este método se implementa la cuantización y esparcimiento (*quantization,sparsification*). Para el primero que es aplicado durante el proceso de compresión del error de proyección, los valores sobre cada dimensión continua del valor original, son aleatoriamente convertidos a un valor discreto, utilizando su norma para reducirlos sobre el rango continuo [0,1] y posteriormente aproximar bajo cierta probabilidad, a su nuevo valor discreto [0,1]. Esta reducción dimensional permite pasar de $n * 32$ bits a solo 32 bits [8]. Para el caso del esparcimiento, este método es aplicado sobre la fase de compresión del gradiente, logrando una reducción de forma más .ªgresiva.ªl seleccionar un subconjunto de valores (ej, 0.1 %) que frecuentemente se representa por un número k de gradientes ortogonales, sobre este método, la técnica con mejores resultados de convergencia es el denominado *TopK*, donde la selección de las k coordenadas mayores (en magnitud absoluta) son tomadas en cada iteración, dejando que el resto de direcciones sean acumuladas hasta eventualmente volverse relevantes"para la dirección de descenso [8].

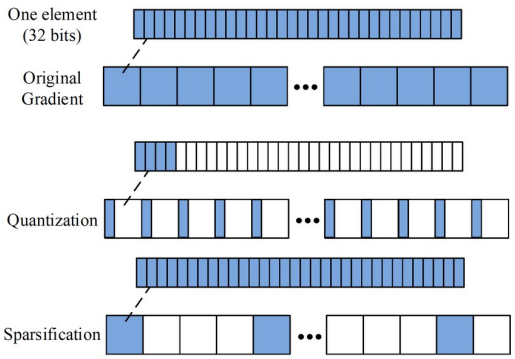


Figura 1: Métodos de compresión utilizados por MicroADAM [8]

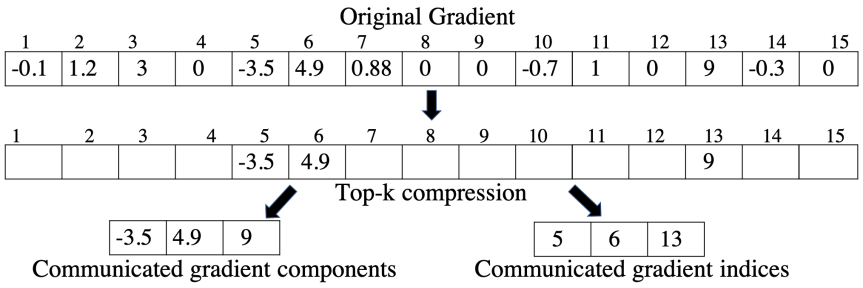


Figura 2: Ejemplo de método Top-K, en la selección de gradientes más importantes

Utilizando como supuestos las expresiones sobre C y Q para los compresores del gradiente y el error respectivamente:

$$\begin{aligned} &||C(x) - x|| \leq q||x||,\ x \in R^d \\ &donde: \\ &q: \text{coeficiente de reducción } 0 \leq q \leq 1 \end{aligned} \tag{1}$$

$$\begin{aligned} &E[Q(x)] = x,\ ||Q(x) - x|| \leq \omega||x||,\ x \in R^d \\ &donde: \\ &\omega: \text{cota inferior de compresión } \omega \geq 0 \end{aligned} \tag{2}$$

En especifico, el algoritmo de compresión *TopK* implementado por MicroADAM logra un valor de $q = \sqrt{1 - \frac{k}{d}}$. A partir de estas cotas, es entonces que para un vector $x \in R^d$ con $\delta = min_i(x_i)$ y $\Delta = max_i(x_i)$, eligiendo una distribución uniforme aleatoria $\xi \sim U[0,1]$, con un nivel de cuantización dado por $u = \frac{\Delta - \delta}{2^b - 1}$, se define $\hat{x} = [\frac{x_i - \delta}{u} + \xi]u + \delta$, como la i-esima coordenada del vector cuantizado $\hat{x}$, y que cumple con la relación:

$$E[\hat{x}] = x,\ ||\hat{x} - x|| \leq \frac{\sqrt{d-2}}{2^b - 1} \frac{\Delta - \delta}{\sqrt{\Delta^2 - \delta^2}} \tag{3}$$

1.2.2. Garantía de convergencia

Delimitando para este apartado la descripción del teorema para las funciones suaves no convexas, se asumen las relaciones dadas anteriormente, así como las siguientes 3 definiciones *estándar* en la literatura de optimización:

1. Sobre una función de perdida $f : R^d \to R$, esta será acotada inferiormente por otra función $f^* \in R$ que es Lipschitz continua (*L-smoothness*):

$$||\nabla f(\theta) - \nabla f(\theta')|| \leq L||\theta - \theta'||,\ \theta, \theta' \in R^d \tag{4}$$

2. Sobre la cota del gradiente estocástico insesgado, si para toda las iteraciones t, el gradiente g_t se encuentra uniformemente delimitado por la constante $G \geq 0$:

$$E[g_t] = \nabla f(\theta_t), ||g_t|| \leq G \tag{5}$$

3. Sobre la cota de la varianza, si para todas las iteraciones t, la varianza del gradiente g_t se encuentra uniformemente delimita por la constante $\sigma^2 \geq 0$:

$$E[||g_t - \nabla f(\theta_t||^2] \leq \sigma^2 \tag{6}$$

Es entonces que bajo estas asunciones, en conjunto con la condición de compresión para el gradiente y error: $(1 - \omega)q < 1$, así como una tasa de aprendizaje $\eta = min\{\frac{\epsilon}{4LC_0}, \frac{1}{\sqrt{T}}\}$, se satisface que:

$$\begin{aligned} &\frac{1}{T}\Sigma_{t=1}^{T} E[||\nabla f(\theta_t)||^2] \leq 2C_0\left(\frac{f(\theta_1) - f^*}{\sqrt{T}} + \frac{L(\sigma^2 + C_2^2 G^2)}{\epsilon\sqrt{T}}\right) + O\left(\frac{G^3(G+d)}{T}\right) \\ &\text{Con las constantes:} \\ &C_0 = \sqrt{\frac{4(1+q_\omega^2)^3}{(1-q_\omega^2)^2}G^2 + \epsilon} \\ &C_2 = \omega q\left(1 + \frac{2q_\omega}{1 - q_\omega^2}\right) \end{aligned} \tag{7}$$

Esto implica que la convergencia contempla la velocidad óptima para los métodos de gradiente estocástico no convexos [3], dada por el término $\frac{1}{\sqrt{T}}$, y que los procesos de compresión y retroalimentación del error solo afectan por el valor de las constantes C_0, C_2 esta velocidad de convergencia.

2. Metodología

Bajo una nomenclatura y estructura del programa similar a la utilizada en el método clásico ADAM, para esta implementación se definen las variables como:

- **f**: función de perdida.
- **d**: tamaño del modelo.
- **k**: densidad del gradiente (*sparsity* = **d** - **k**).
- θ_t: parámetros del modelo.
- g_t: gradiente del modelo.
- **t**: iteración.
- η: tasa de aprendizaje.
- λ: coeficiente de atenuación de los pesos.
- m_t, v_t: primer y segundo momentos del gradiente.
- e_t: vector del error de retroalimentación.
- **b**: bits de cuantización para el error.
- **m**: tamaño de la ventana deslizante para compresión *sparse*.
- **G(I,V)**: ventana de dimensión $m \times k$ para los indices I y valores de gradiente V, del método Top-K.

En el flujo mostrado sobre el pseudocódigo (1), se observa la inicialización de los momentos y error de retroalimentación en ceros, tras la entrada al ciclo iterativo, se realiza la estimación del gradiente y posteriormente la corrección con la retroalimentación del error (aplicando la cuantización inversa). En la siguiente instrucción, se aplica el método de compresión del gradiente retroalimentado (α) utilizando la técnica Top-K, como en este paso se toman los indices y valores de los gradientes más relevantes a modo de buffer, entonces será necesario descartar estos .ºutliers"de α dejando el error de reducción. En los pasos 10 y 11 se calculan los parámetros para la cuantización y posteriormente son aplicados para la compresión del error, en seguida los valores e indices de la ventana G son actualizados y finalmente este buffer es utilizado para para calcular de manera dinámica los parámetros çlásicos"de ADAM, a través de la función *ADAMSTATS*.

Algorithm 1: Pseudocodigo de MicroADAM con cuantización (compresión) de error de retroalimentación (EF)

1 **Function** *MicroADAMef(β_1, β_2, ϵ, G, T, d, k)*:
  Data: $coeficientes\ de\ momentos\ :\ \beta_1, \beta_2\ |constante\ estabilidad:\ \epsilon|ventana\ deslizante:\ G|dimension\ de\ modelo:\ d\ |densidad\ de\ gradiente:\ k$
2 $m_0, v_0 \leftarrow 0_d, 0_d$;
3 $\delta_1, \Delta_1 \leftarrow 0, 0$;
4 $e_1 \leftarrow 0_d^{4b}$;
5 **for** t *in* T **do**
6 $g_t \leftarrow \nabla_\theta f(\theta_t)$;
7 $a_t \leftarrow g_t + Q^{-1}(e_t, \delta_t, \Delta_t)$ ▷ Cuantización inversa (error feedback);
8 $I_t, V_t \leftarrow T_k(|\alpha_t|)$ ▷ "Sparsification"del gradiente;
9 $\alpha_t[I_t] \leftarrow 0$;
10 $\delta_{t+1}, \Delta_{t+1} \leftarrow min(\alpha_t), max(\alpha_t)$;
11 $e_{t+1} \leftarrow Q(\alpha_t, \delta_{t+1}, \Delta_{t+1})$ ▷ Cuantización;
12 $G_{i,:} \leftarrow (I_t, V_t)$;
13 $\hat{m}_t \leftarrow AdamStats(\beta_1, G)$;
14 $\hat{v}_t \leftarrow AdamStats(\beta_2, G^2)$;
15 $\theta_{t+1} \leftarrow \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$ ▷ Actualización de parámetros;

Los cálculos en el pseudocódigo (2) contemplan la asignación del momento z_t a partir de los valores del gradiente contenidos en la ventana sparse $G(I, V)$, en este punto se estará trabajando con un porcentaje de los valores (ej. $k = d/100 = 1\,\%$ de densidad de valores) que agilizará la operación especialmente sobre un modelo de datos considerablemente grande. Para r, que será el exponente de los factores β calculado dinámicamente, se contempla el número de iteración (t), la i-esima fila de la ventana de $G(I, V)$, y el tamaño de esta estructura (m), dado que G funciona como un buffer circular de gradientes principales, los valores agregados recientemente tendrán un valor $r = 0$, mientras los que llevan más iteraciones tendrán un exponente mayor $r = m - 1$.

Algorithm 2: Pseudocodigo de ADAM Statistics

Function *ADAMSTATS*(β, $\mathcal{G}$, t, m, d)**:**
  $z \leftarrow 0_d$;
  for $i \in \{1, 2, \dots \text{mín}(t, m)\}$ **do**
    $r \leftarrow (t - i - 1)\,\%m$;
    $z[\mathcal{I}_i] \leftarrow z[\mathcal{I}_i] + \beta^r \mathcal{V}_i$;
  return $\frac{(1-\beta)z}{1-\beta^r}$;

Algorithm 3: Pseudocodigo de Cuantización

Function $Q(x,\ \delta,\ \Delta,\ b)$**:**
  $u \leftarrow \frac{\Delta - \delta}{2^b - 1}$;
  $x_Q \leftarrow \lfloor \frac{x-\delta}{u} + \frac{1}{2} \rfloor$;
  return x_Q ;

Algorithm 4: Pseudocodigo de Cuantización Inversa

Function $Q^{-1}(x_Q,\ \delta,\ \Delta,\ b)$**:**
  $u \leftarrow \frac{\Delta - \delta}{2^b - 1}$;
  $x \leftarrow x_Q \cdot u + d$;
  return x_Q ;

3. Experimentos

La implementación del método puede ser consultada en el repositorio: `https://github.com/L4rralde/MicroAdam`

Las ejecuciones fueron realizadas en un equipo bajo el soporte de una CPU Apple silicon M1 y 8 GB de RAM. Empleando 2 conjuntos de datos populares importados desde la librería de python *Scikit-learn*, el primero de ellos **Iris** [1], con 3 clases, 150 instancias y 4 características, mientras el segundo conjunto con una mayor complejidad, "Breast Cancer Wisconsin (Diagnostic)"[2], contiene 2 clases, 569 instancias y 30 características.

3.1. Modelos de clasificación con MicroADAM como optimizador

Configurando nuestra implementación del optimizador para un modelo de clasificación en *Scikit-learn*, se evaluó el desempeño para el conjunto "Breast Cancer"sin incluir la retroalimentación del error de compresión, optimizand con 3 casos: sin la compresión del gradiente, eligiendo el valor para $k = \frac{d}{100} = 1\,\%$ *density*, y $k = \frac{d}{10} = 10\,\%$ *density* (fig. 3).

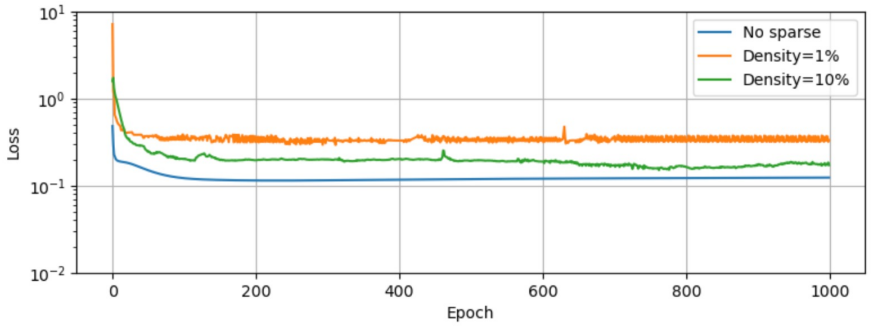
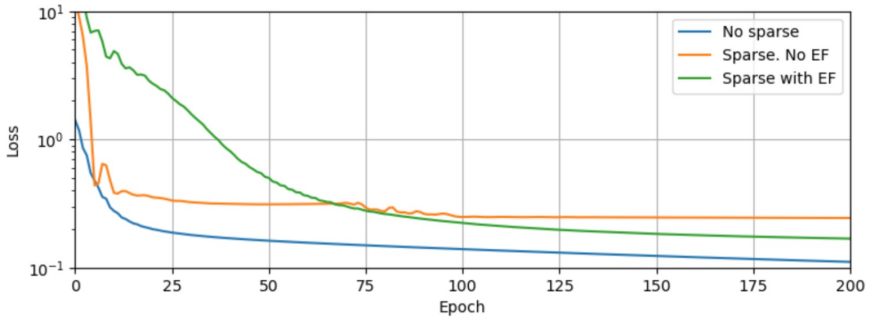
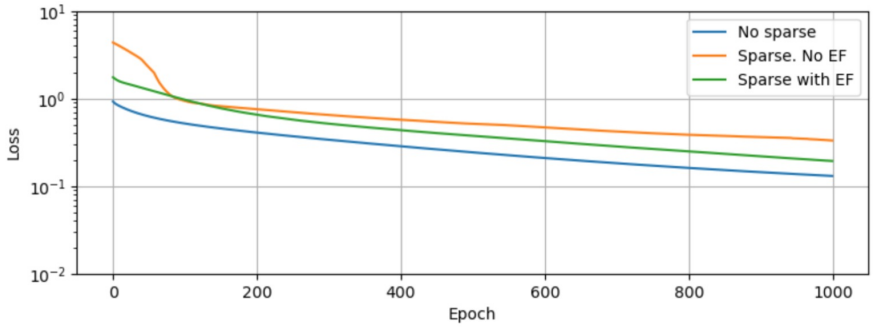


Figura 3: Minimización del error de entrenamiento para el conjunto *Breast Cancer* con valores k diferentes.

Se puede observar cierta inestabilidad en la convergencia que es periódicamente centralizada para una densidad del 1 %, pero algo más aleatoria para el 10 %. En la figura 4 se presentan los resultados para ambos conjuntos de datos, esta vez realizando una comparación del efecto que tiene la retroalimentación del error de compresión (EF), manteniendo el porcentaje de densidad al 1 %.



(a) Conjunto de datos *Breast Cancer*.



(b) Conjunto de datos *Iris*.

Figura 4: Minimización de error de entrenamiento con retroalimentación y densidad 1 % (rojo), sin retroalimentación y densidad 1 % (amarillo), sin retroalimentación y sin compresión de gradiente (azul).

Al incluir el error de proyección en la estimación de los gradientes, se obtiene una mejora con respecto al experimento basado unicamente en la reducción del gradiente, el resultado asintótico para el caso con retroalimentación en ambos conjuntos parece ser alcanzado cerca de la misma época de entrenamiento, siendo igualmente en ambos casos el punto existe una perturbación en la tendencia del experimento sin retroalimentación, aunque no alcanza el mismo comportamiento que el modelo base (linea azul), en teoría debería conseguirse una menor inversión de memoria.

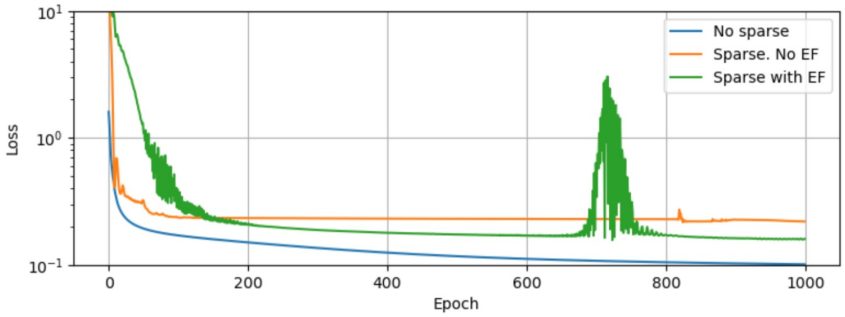
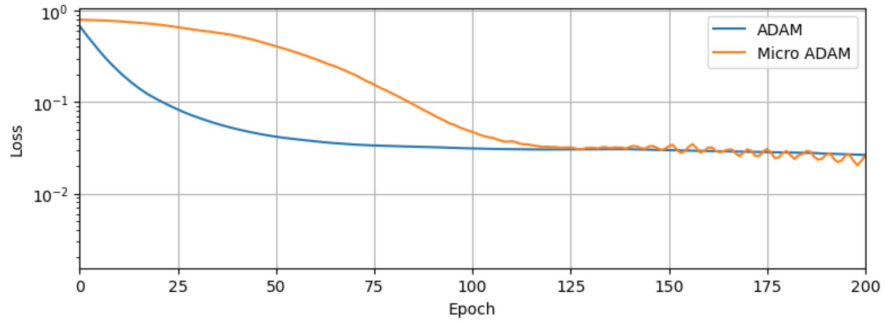


Figura 5: Perturbación de convergencia sobre el conjunto *Breast Cancer* al incrementar la *sparsity*.

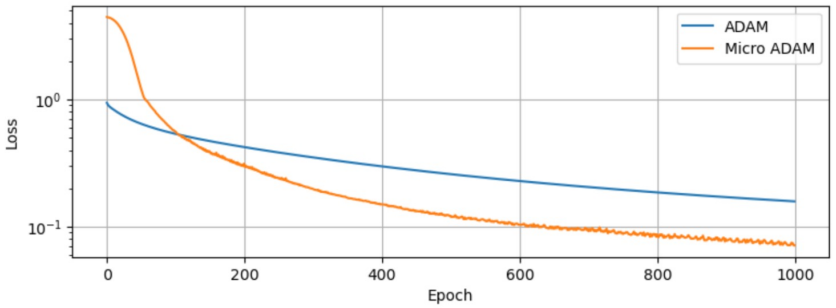
Probando el caso donde la densidad del gradiente disminuye (fig. 5), esto provoca que la corrección de error por la retroalimentación se vuelva inestable. Esto podría asociarse con la probabilidad de que δ sea muy similar a Δ, resultando en insuficientes elementos a cuantizar, y dado que este procedimiento utiliza una división sobre $u = \delta - \Delta$, entonces esto induciría valores inestables.

3.2. Comparación ADAM y MicroADAM

Al comparar la implementación de ADAM y MicroADAM (fig. 6), se establecieron los mismos parámetros comunes: $\beta_1 = \beta_2$, y el resultado es una aparente mejora en la convergencia de MicroADAM, sin embargo al incrementar el número de épocas, las inestabilidades en esta implementación logran incluso divergir, contrastando con la tendencia de ADAM.



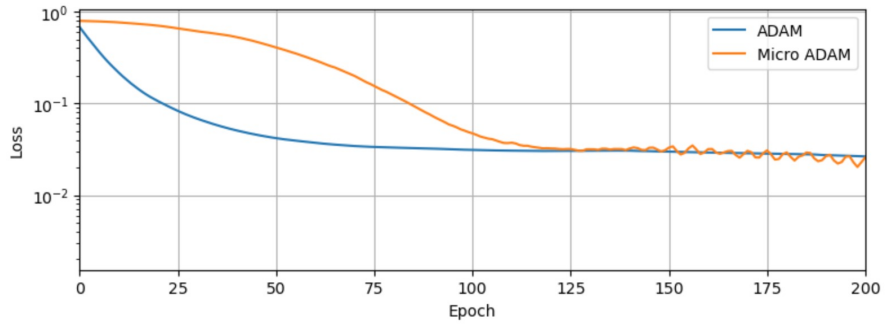
(a) Conjunto de datos *Breast Cancer*.



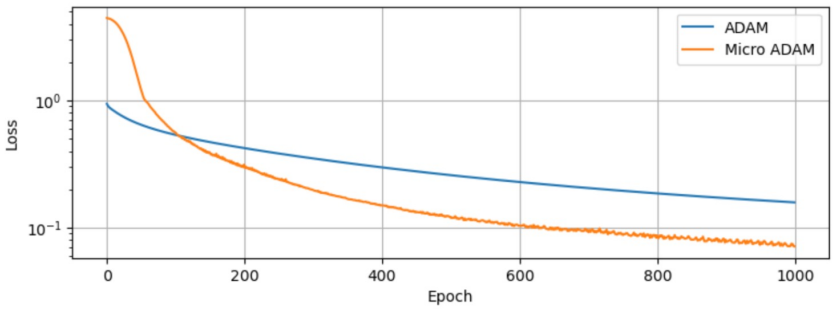
(b) Conjunto de datos *Iris*.

Figura 6: Comparación de convergencia en entrenamiento con $\beta_1 = \beta_2$ entre las implementaciones propias del método ADAM clásico (azul) y MicroADAM (amarillo).

El comportamiento se vuelve mucho más inestable si tomamos valores sin normalizar (fig. 7.a), esto como resultado de una nula representación del error de retroalimentación por la técnica de cuantización, sin embargo, si se modifica la resolución de cuantización a 8 bits (fig. 7.b), entonces este comportamiento oscilante se reduce en gran medida.



(a) Conjunto de datos sin normalizar.



(b) Conjunto de datos con resolución de cuantización a 8 bits.

Figura 7: Comparación de convergencia en entrenamiento del conjunto de datos *Breast Cancer*.

4. Conclusiones

Aunque en el presente trabajo no pudo ser evaluada la reducción de los requerimientos en la memoria que ocupa el método ADAM tradicional, principalmente por falta de hardware que permita entrenar algún conjunto de datos a nivel de optimización distribuida, las evaluaciones mostradas para los modelos a menor escala permitieron evidenciar una de las principales desventajas de MicroADAM, la cual es la falta de robustez como resultado de los múltiples parámetros. Como se describió en las figuras 6 y 7, la convergencia del optimizador se ve fuertemente afectado por la calidad de la configuración de G como ventana de gradientes dispersos, así como de la resolución utilizada por el cuantizado para el proceso de retroalimentación. Adicionalmente, aunque no se evaluó explícitamente en este trabajo, la diferencia en las velocidades de convergencia entre el modelo ADAM y MicroADAM, era bastante en favor del método clásico, lo que sería otra característica de interés en futuras comparaciones.

Referencias

[1] R. A. Fisher, *Iris*, UCI Machine Learning Repository, DOI: https://doi.org/10.24432/C56C76, 1936.

[2] W. William, M. Olvi, S. Nick y S. W., *Breast Cancer Wisconsin (Diagnostic)*, UCI Machine Learning Repository, DOI: https://doi.org/10.24432/C5DW2B, 1993.

[3] S. Ghadimi y G. Lan, *Accelerated Gradient Methods for Nonconvex Nonlinear and Stochastic Programming*, 2013. arXiv: 1310.3787 [math.OC]. dirección: https://arxiv.org/abs/1310.3787.

[4] D. P. Kingma y J. Ba, *Adam: A Method for Stochastic Optimization*, 2017. arXiv: 1412.6980 [cs.LG]. dirección: https://arxiv.org/abs/1412.6980.

[5] N. Shazeer y M. Stern, *Adafactor: Adaptive Learning Rates with Sublinear Memory Cost*, 2018. arXiv: 1804.04235 [cs.LG]. dirección: https://arxiv.org/abs/1804.04235.

[6] T. Dettmers, M. Lewis, S. Shleifer y L. Zettlemoyer, *8-bit Optimizers via Block-wise Quantization*, 2022. arXiv: 2110.02861 [cs.LG]. dirección: https://arxiv.org/abs/2110.02861.

[7] J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar e Y. Tian, *GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection*, 2024. arXiv: 2403.03507 [cs.LG]. dirección: https://arxiv.org/abs/2403.03507.

[8] X. He, F. Xue, X. Ren e Y. You, "Large-Scale Deep Learning Optimizations: A Comprehensive Survey," *arXiv (Cornell University)*, Jan. 2021. DOI: 10.48550/arxiv.2111.00856.